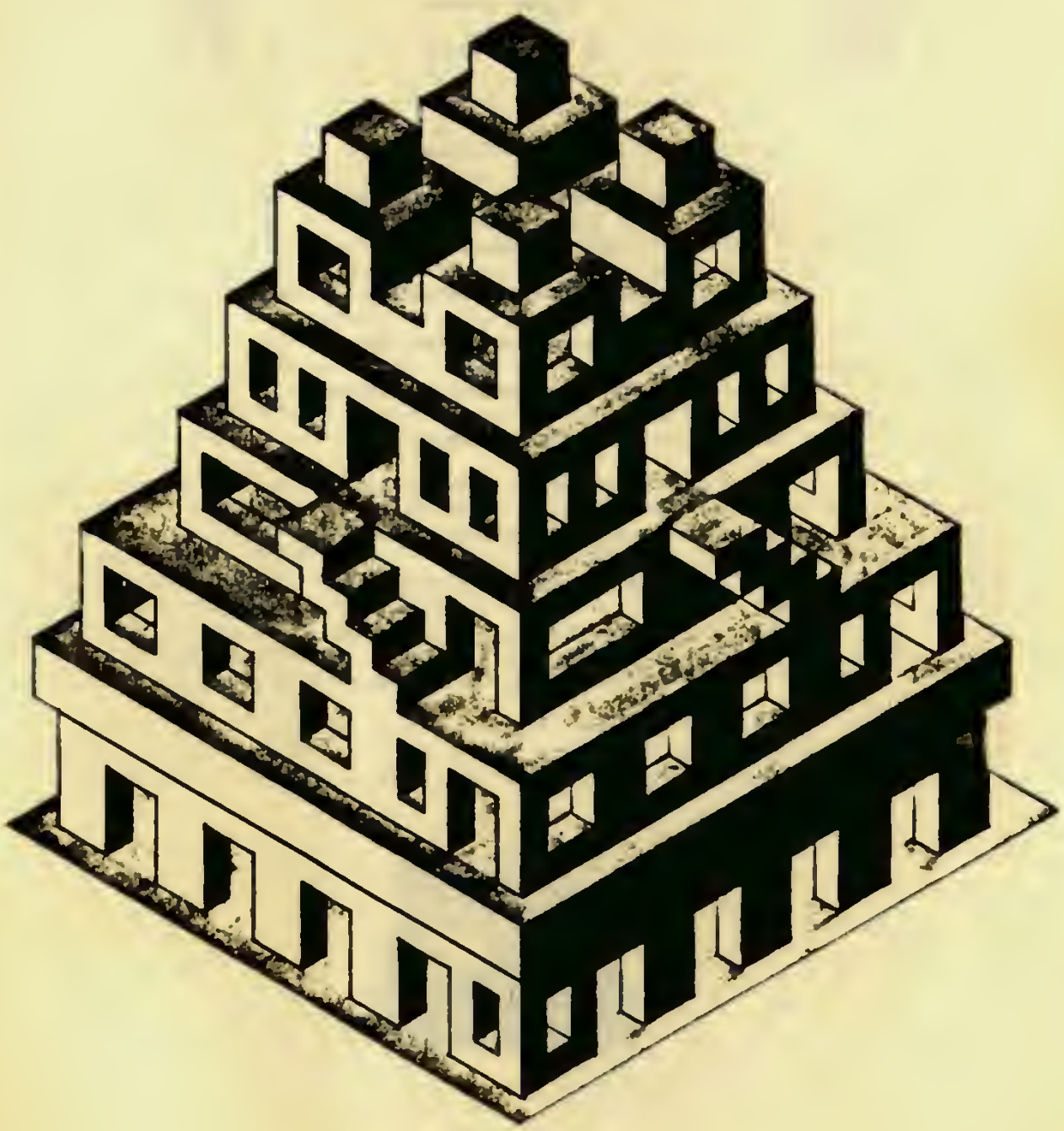
Adventure Coder

AUGUST 1989 ISSUE 2 £1



CONTENTS

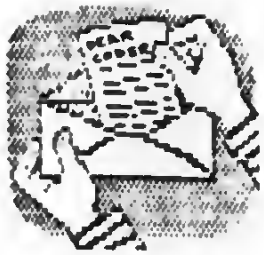
Long Hot Summer

One things for sure this month - it's too hot to compute! I've even been hearing tales of people who actually put their computers away the whole summer! So writing an adventure game during the summer months might seem crazy, but that's just what a lot of writers have to do. How else do they get their games ready for the end of the year! It's generally acknowledged that sales are lower in the summer and at their highest before Christmas. But what can you do about the weather? Unfortunately Britain is still a poorly prepared country when it comes to dealing with heatwaves. Let's face it, really hot days are something of a rarity this north of the equator, and we're lucky to get a full week's swelter, let alone a months, whereas places such as Greece roast all year long in temperatures double what we get here. In America, they get hotter weather than us too, and for longer periods, so they know exactly how to deal with it. We don't. How many houses do you know with air-conditioning? Fans in the ceiling, or on a shelf? British cars that have air-conditioning and anti-glare strips fitted as standard on all models? Us Brits have to cope with hot days as best we can - the rest of the year we're okay - out with the wooly jumpers and up with the central heating. Winter's ideal for computing, but summer? Pass me another ice cream... Is anyone playing adventures now? If not, they certainly are writing them, no matter what the weather. I've cut down on use of my computer to the barest minimum recently, and abandoned using it in the bright sunny afternoons altogether. I always hate having to draw the curtains right across otherwise, just to cut down enough sunlight to be able to merely read the screen! I've tried two bedrooms and neither solve the problem. Anyway, the sweat makes my glasses slip down, so I prefer to compute in the evening when it's dark. Right now though, it's the evening, but it's probably hotter than it was during the day! No matter, the magazine must continue! And so it does, with the second issue you have here! You might find it too hot to play adventures, but you can always take this outside and read it in the garden, with a glass of coke at your side!

This issue sees the start of articles on STAC, as well as more on GAC, PAW and adventuring in general. There's an upoated list of useful addresses and utilities available, plus two crosswords. Two! Ahem. Apologies are in order over the unreadable clue numbers in last month's crossword. I've reprinted it for you to try properly this month, but incase you struggled to solve it anyway, there's a second crossword to try, a little easier than the first. Take your pick! Also, I've abandoned the inverse page numbers, you'll be glad to hear! Hopefully this issue looks better than ever!

Finally, can I welcome our overseas readers in France, Norway and Denmark to the magazine - it's great to know that Adventure Coder is now worldwide!!

Letters Page



"Our aim is to re-vitalise the 8-bit adventure scene" - *so says Mitch Pomfret of M.S.B Games in our first letter this month:-*

Dear Coder,

As you may know, M.S.B GAMES run a successful basketball Play-By-Mail (PBM) game, entitled "Slam Dunk". Although we are continuing our PBM operations, with the release of a Motor Racing PBM and a Football PBM, we are also extending our interests into computer software. Most of our computer software development is going into writing adventure software and we have created a department, Storyboard Designs, to deal with our adventure developments.

To begin with, we are writing our adventures using GAC. The reason for this is that our adventures are being designed and written by expert adventurers, whose speciality lies in writing the text and puzzles, as opposed to being able to code. We hope, however, to take these GAC created games and have Graphics Programmers add graphics, giving the games that vital sales appeal.

Our first project has been written 'in house' and is a five part horror adventure, entitled "Blood Of A Vampire". We are currently planning the graphics for the game, as well as discussing the publishing of the game with several larger software houses.

Our aim is to re-vitalise the 8-bit adventure scene, which is slowly being moved to be replaced by 16-bit 'wonder games'. We do plan to write 16-bit adventures, but 8-bit games are equally as important! Eventually we hope to set up our own 'Adventure Development House', with 'in house' and freelance programmers/adventure writers working together, to produce the adventure games of the future, along with the likes of Magnetic Scrolls and Infocom! Getting to that stage, however, involves a lot of hard work, from dedicated game writers/coders.

To help us accomplish the above task, we need to recruit adventure writers/coders, to work on their own ideas either 'in house' or freelance. To begin with, these games will be published by an established company, slowly being phased out in favour of our own publishing operation...

Mitch Pomfret, M.S.B Games, 2 Bude Close, Bramhall, Stockport, Cheshire, SK7 2QP.

Why not write to them - perhaps you've some experience in adventure writing, or have an idea for a game?

Dear Editor,

I'm pleased to see that your first issue met my expectations. I enjoyed it immensely, and it should improve even further once we can see articles from other readers.

As an adventure writer myself I hope your magazine will be a constant source of information and help for me. The addresses of magazines, companies and stationery companies were all

very useful as I have just released my own adventure.

I hope to see many articles on using the PAW system, as this is easily the best system for the Spectrum. If Glisoft are reading this, when will you release an ST PAW! That would be something to look forward to, for both writers and players.

I agree with all the people who say they are sick of people constantly prophesising 'the death of the text adventure' - certain glossy magazines especially. I prefer text-only adventures on computers such as the Spectrum, and other 8-bits. On the 16-bits graphics can be used to much better effect, such as the Magnetic Scrolls games. I dislike the roleplaying games currently in favour and these are nothing to do with adventure games. One of my favourite adventures is Leisure Suit Larry II, which uses the ST's graphics to great effect.

Well, good look with your magazine and I hope you run at least as long as Adventure Probe.

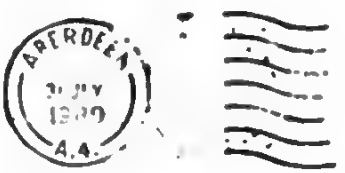
Mental Image, Patrick Walsh, Berkshire.

I've come across many adventurers that still prefer text-only adventure games. Why then do we get only graphic adventures from the big companies? The reason is that shops won't stock an adventure unless it has pretty pictures.

As for further articles on the PAW, rest assured that they'll continue to flow from our PAW columnist for a long time yet! This utility so far seems to be the favourite amongst readers. Anyone out there dislike it?

As for an ST PAW, you never know... the-he!)

Snippets



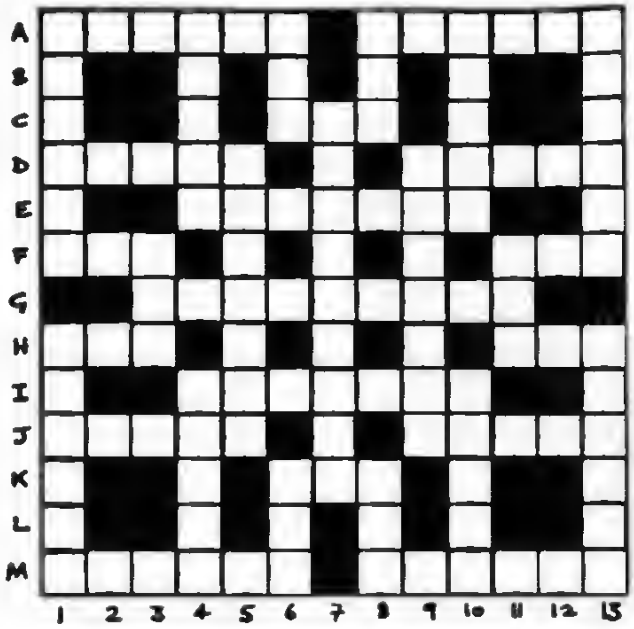
"Issue 1 is great! PAW Prints is excellent!!" - Darren Rose, Norfolk.

"Issue 1 was great - more power to your elbow!" - Gordon Inglis, Edinburgh.

"I've just got the first issue of 'Coder' and very good it is too." - Gerald Kellett, Stamford.

"Issue 1 was gratefully received and provided interesting reading." - Paul Brunyee, Leicester.

"I found it very interesting reading as I am not very good at writing adventures so it will be a big help to me." - J Bailey, Tyne & Wear.



Crossword #1

by C.Hester

VERTICAL CLUES

A1) Forbidden Iniocom title in Autumn? (6)
A4) Home of the Three Bears and more (5)
A6) Neptune's delight (3)
A8) Deity (3)
A10) Saddest and loudest person in the town (5)
A13) Fabled beastie turned computer (6)
C7) Angry Bomb The Bass hit without the Beat? (9 or 3,6)
D5) Ask further information (7)
D9) Obstacle (7)
F3) Rod Hull's infamous bird (3)
F11) D+D genre taking over from adventures? (1,1,1)
H1) Method (6)
H13) Crashing icon (6)
I4) Teach skills on the rails: (5)
I10) Thick (5)
K6) All objects are red herrings without a --- (3)
K8) Novel prefix (3)

HORIZONTAL CLUES

A1) Examines closely copies of popular adventure mag (6)
A8) Written with incentive (6)
C6) It's a plus with Maths (3)
D1) Absence of the singular (2,3)
D9) Existing creature (5)
E4) Lowest pointed mark (3,4)
F1) Finish together a knot (3)
F11) LOAD before athletics (3)
G3) Agatha Christie play for doomed peripheral? (9)
H1) Very small amount of money - one p short to buy a hot bowl? (3)
H11) Easy talkative American petrol (3)
I4) No exits! (7)
J1) Check this for progress (5)
J9) Kingdom (5)
K6) Jar for ashes (3)
M1) Average spoilsport in The Sentinel? (6)
M6) School exam for Ingrid's creatures minus 8. (1,5)

GAC POKEs

FOR COMMODORE 64 GAC USERS ONLY by Christopher Hester

When I first got GAC for my 64 I was annoyed to find it used a constant dark blue background colour for both the screen and the border. This isn't so bad until you've spent several hours working with GAC reading the cyan and yellow text colours used. Cyan on dark blue is acceptable, whereas yellow is pushing the limits of fuzziness a bit. As for the green used at the base of the Graphics screen whenever a command is used, this is hopeless at highlighting the text, as Incentive intended. Green on dark blue! Yeuch!!! Fuzzy even on a Commodore monitor, though a majority of GAC users will be using their home TV sets where all colours are not as clear as you'd get with a monitor. Programmers don't seem to realise this, and quite happily put in all kinds of disgusting and unreadable colour combinations in their programs - well they can read them on their monitors!

Besides my dislike of the blue background there's a couple of good reasons why it shouldn't be used. Firstly, any colour left constantly on the screen in large amounts affects the health of your TV set. The brighter the colour, the worse the effect. I'm particularly annoyed by games that use strongly coloured borders the whole time they're on - if the border colour changes then that's okay. The reason is that TVs are not meant to display static colours at all, though with so much recent computer graphics visible on TV, I feel sure manufacturers are having to allow for this a bit now. Your older TV sets, however, require a constant change of colours, as that is what you get in a typical TV programme. Remember any programmes where the border stays the same colour throughout the show: No, they don't do that. Also, TV images use the whole range of TV colours, not the small palette you get of ultra-bright colours with home computers. These colours are therefore unnatural for the TV tube to display for lengthy periods, unless they are kept changing. Imagine then the effect even a dark blue border has if left on the screen for hours on end as you type in your GAC code.

The second reason not to use a constant background of colour in GAC is that every piece of text, no matter what colour used, looks far, far superior if you remove the dark blue and use *black* instead. Black removes *all* the fuzziness from whatever text colour you use! Green that appeared yeuchy on dark blue suddenly becomes the sharpest of colours; cyan is similarly clear. Also, with black, the border isn't lit, nor vast quantities of the screen whilst you enter one line of code at the top - why colour all that space? Black is better for your TV and your eyes! What a shame then that Incentive chose a dark blue screen for GAC. If only you could alter it to black.

Well you can, and I'm here to show you how! Not only that, but I'll leave it open to you to choose any colour you wish, not just black - you may like a dark brown screen, or dark grey, whatever! To change the dark blue colour of the GAC screen, you must resort to POKEs. If you've never POKEd a game before, don't worry, it's really quite easy, and there's nothing illegal about it at all. It's only illegal when you POKE a game in order to copy it. Note that this article is only of use for Commodore 64 GAC users too - my POKEs won't work at all for the Spectrum GAC!

Firstly, are you using a tape or a disk copy of GAC. If you're using a tape copy, then you'll need a Reset Switch at the back of your computer. If you haven't got one, they're only a fiver - look in the small ads of the computer magazines. These enable you to Reset the computer, taking control away from the program and returning you back to BASIC, but leaving the program intact. Owners of a disk copy of GAC

might also wish to use a reset switch, or follow the method for disk users below.

Tape users: LOAD your copy of GAC and Reset the machine. You should now be back at the blue start-up screen as if you'd just turned on the computer, with the cursor flashing.

Disk users: insert your GAC disk into your drive and type LOAD "GCODE",8,1 <RETURN>. The main body of GAC should now load but not run.

From here, both tape and disk users can follow the same directions. I've compiled a list of all required POKEs to alter the screen colours in GAC. I found though that you need more than one POKE to achieve this, as the screen colour is set several times in GAC. To remove all the dark blue colouring used, follow the POKEs below, noting that N denotes your NEW choice of colour. The range of colours on the Commodore 64 is as follows:-

0 - black	8 - orange
1 - white	9 - brown
2 - red	10 - pink
3 - cyan	11 - dark grey
4 - purple	12 - gray
5 - green	13 - light green
6 - dark blue	14 - blue
7 - yellow	15 - light gray

So if you wish to alter the dark blue to dark gray, N will equal 11. Unfortunately to alter the text colours used in GAC requires so many fiddly POKEs that I've omitted that option - it is assumed that cyan and yellow are okay for the text. The reason is that GAC splits a lot of text up into words that start with a yellow capital then change to cyan. Plus the colours aren't stored in a normal machine code way, but in a GAC coded way that adds to the confusion. I think you're best settling for black here, as it suits cyan and yellow text well, so choose N to be 0. Here then are the pokes:-

POKE 26208, N = border colour

POKE 31981, N = graphics base colour

POKE 39783, N+48 = Main Menu background colour

POKE 40144, N+48 = Edit Verbs background colour

POKE 40453, N+48 = Edit Adverbs background colour

POKE 40478, N+48 = Edit Nouns background colour

To use these POKEs, simply enter them in BASIC. So choosing N = 0 for a black screen, the first POKE would come out as POKE 26208,0 <RETURN>. Then, when you're happy with your choices, you can set GAC running by entering this:-

SYS 32768 <RETURN>

GAC+ POKEs

If you've recently upgraded to GAC+, however, you'll find the previous POKEs no longer work, but not to worry, I've converted them for you. Here then are the same POKEs as before, but for GAC+.

Insert your GAC+ disk and enter LOAD "GCODE",8,1. Now choose the POKEs required:-

POKE 25367, N = border colour

POKE 31176, N = graphics base colour

POKE 39230, N+48 = Main Menu background colour

POKE 39591, N+48 = Edit Verbs background colour

POKE 39900, N+48 = Edit Adverbs background colour

POKE 39925, N+48 = Edit Nouns background colour

SYS 32768 = start GAC+.

And there you have it, I've certainly found both set of pokes invaluable and made my own personal backups of GAC and GAC+ with a black background and border used.

Perhaps you've come up with similar POKEs for your different computer's version of GAC? If so, send them in! Better still, has anyone come up with any POKEs to improve GAC in any way: display words instead of text in the code, or move the graphics screen up for more text underneath?

Sean Ellis, writer of GAC

Fish Chart

by C. Hester

1) "It's A Fin" - The Perch Shop boys
2) "Born To Swim" - Bruce Squidbreem
3) "U Got The Hook" - Prawnce
4) "Owner Of A Lonely Trout" - Dace
5) "Fixing A Bowl" - The Beatfish
6) "Tubular Gills" - Pike Oldfillet
7) "Coddly Toy" - Roachford
8) "Rock Around The Tank" - Gill Haley & The Surfers
9) "Heaven Knows I'm Mackerel Now" - Monkfishe.
10) "On The Tail Of The Lonesome Pike" - Laureel & Perch

PAW Prints

by George E March, 93 Roberts Street, Newcastle-Upon-Tyne. NE15 6BE.

This month I thought I'd like to try and explain to the new (or even experienced) P.A.W writer, some of the more unusual and interesting abilities of its parser and vocabulary tables.
For instance, any verbs (or conversion nouns, ie, a noun that can be used as a verb if there's no verb given in a sentence) with a number less than 14 are taken as directions (eg, north and enter), any nouns with a number greater than 19, but less than 50 are taken as proper nouns (those names of people and place names, etc), and those with a number greater than 49 are names of objects, etc, but what about those with a number greater than 13, but less than 20? They're still conversion nouns, and so can still be taken as a verb, if no verb is available, but are not directions!
For example if we give a word (that we wish to be both a verb and a noun (eg, 'plant'),the noun number 15, and also make sure that there is nc verb with the same number, give a second noun (eg, begonia),the number 52, and then give a verb we need to act as an interpreter (eg, bury) the number 39, it allows us to create a routine such as..

```
PLANT * LET 33 39

BURY PLANT PRESENT 4 SYSMESS 61 PLACE 4 255 SWAP 4 5 ANYKEY DESC
```

Using object 4 = 'A small pot plant', object 5 to be 'A plant buried in some soil' and sysmess 61 to be 'You plant the plant into some compost'. Which now allows us to PLANT the PLANT if it were present, into the ground where we stand, then the routine swaps the plant in a pot for a plant in some soil, and also by changing the noun number in an examine routine, we can both examine the plant and the begonia..

```
X PLANT LET 34 52
```

Which changes the number of the noun put in by the player, into another, to become.

```
X BEGONIA PRESENT 4 SYSMESS 60 MESSAGE 50 DONE

X BEGONIA PRESENT 5 SYSMESS 60 MESSAGE 51 DONE
```

With sys' 60 being 'It's a lively, green begonia, with beautiful pink flowers, ' (remember the space!), message 50 being 'but it's feeling a bit cramped in its too small pot!' and message 51 being 'sitting comfortably in some compost!'. So we can PLANT the PLANT, which, with some slight alteration can become BRIDGE the gap, and cross the BRIDGE, etc, and instead of two, seperate routines giving the same answer, eg..

```
X BRANCH AT 2 MESSAGE 27 NEWTEXT DONE

SEARCH BRANCH AT 2 MESSAGE 27 NEWTEXT DONE
```

Using location number 2 as a tree grove and message 27 to read 'The branches are far too high to reach..', with a newtext to break off any input after something that can't actually be done, we could use instead..

```
X BRANCH AT 2 LET 33 43

SEARCH BRANCH AT 2 MESSAGE 27 NEWTEXT DONE
```

Which thus converts the word examine (or X) into the word, search, to become the routine coming after it! Or we could even do away with the search and examine bits altogether!

```
* BRANCH AT 2 MESSAGE 27 NEWTEXT DONE
```

Which works with any verbs at all (for example, GET branch, PULL branch, etc), and now onto that combat sequence I promised you last month. Now there are a few ways of doing this, so I'll detail just two of them for space, though the 'calling' routine for both is nearly the same, with possibly a routine in response to start the ball rolling. For example our first type of combat routines could be started by..

```
ATTACK ? PRESENT 4 CLS LET 51 4 MESSAGE 63 PROCESS 6 PROCESS 7 DONE
```

Which, if the player wishes to attack an opponent (who is in the same location), the routine clears the screen of text/graphics, prints a relative message, using LET 51 4, which replaces any underlines in text, with the name of the object number specified, or alternatively in process 1.

```
* _ PRESENT 4 CHANCE 10 CLS LET 51 4 MESSAGE 63 PROCESS 6 PROCESS 7
DONE
```

Now you can see that I've used object number 4 as our opponent (the name is upto you), and I've used message 63 to read 'You attack the _', and process 6 as our combat result process. Process 7 prints out the appropriate messages resulting from any combat, and in this first combat example, I've also used a couple of random flags in process 1..

```
* * RANDOM 80 RANDOM 81
```

These need not be in the * * ZERO 31 option which I'll mention later, with flag 80 being the players strength, and flag 81 the opponents, we'll also need a third, blank flag as a 'counter' for an on/off effect, which I've taken as flag 82. Now for process 6 (our combat process) in this first example, we have the same flag measuring routine as I detailed in last months 'Coder'. Now we could also provide the player with a weapon to use in the game (I've given examples of two), with object 10 being 'A heavy broadsword' and obj' 11 being 'A boot knife', and then at the beginning of process 6 we could have..

```
* * 0 CARRIED 10 PLUS 80 15

* * 1 CARRIED 11 PLUS 80 7
```

Which then adds 15, or 7 to the flag 80 measurments below..

```
* * 2 CLEAR 82 SAME 80 81 NOTDONE
```

Which, if the players and his/her opponents strength levels are the same, won't do anything, and goes back to the calling routine in response, which moves the routine onto process 7 (the results messages), and then back onto process 1, and starts the random number sequence off again..

```
* * 3 LET 82 1 SUB 80 81 ZERO 81 NOTDONE
```

But, if they're not the same, the routine above first makes flag 82 (our counter) equal to one, then takes the contents of flag 80 away from flag 81 , and if 81 is nothing, then 80 is greater than 81, otherwise..

```
* * 4 LET 82 2 NOTDONE
```

82 becomes 2, and thus if flag 82 is nothing, then no one is hurt, if 82 equals 1 then 80 (and therefore the players strength) is greater than 81, and if 82 is 2, then 81 is more than 80! Okay so far? (No! - Ed.)

And so onto process 7, which is visited after the flags are measured, and firstly we have to decide what to do if either the player, or their opponent dies? For this we'll need another object to become the opponents dead body, which I've used as object number 5, again any names are upto you..

```
* * 0 ZERO 81 NEWLINE MESSAGE A SWAP 4 5 ANYKEY DESC

* * 1 ZERO 80 NEWLINE MESSAGE B NEWLINE TURNS NEWLINE SCORE NEWLINE END
```

And so if flag 81 (the opponents strength) is zero, then the opponent is dead, and the routine swaps our live opponent for a dead one, but if flag 80 (the players strength) is nothing, then it prints a message saying so, turns taken, the score, etc, and ends the game for a restart. Now we also need a way of telling the player exactly what's going on, and so in process 7 we'll also need three little routines to measure just that..

```
* _ 0 ZERO 82 MESSAGE C NEWLINE PROCESS 6 PROCESS 7
```

So if 82 is nothing, then no hits are scored, so message C = 'You both miss!', it prints a space between this message and the next (makes it look cleaner) and returns to the beginning of process 6, and does it all over again, until one of them is dead..

```
* _ 1 EQ 82 1 MINUS 81 10 MESSAGE D NEWLINE PROCESS 6 PROCESS 7
```

For the routines given above, if flag 82 is 1, then 80 is more than 81, with message D being 'The opponent is wounded', though I've left out the LET 51 4 this time..

```
* _ 2 EQ 82 2 MINUS 80 10 MESSAGE E NEWLINE PROCESS 6 PROCESS 7
```

Otherwise, if 82 is 2, then the player is hurt, his/her strength is reduced and message E printed, and then back to the beginning again..

Now for this second combat example (which I've designed for a 128 gamer) in response we'll need a 'calling' routine nearlly the same as our first version, but without process 7 being called..

```
ATTACK ? PRESENT 4 CLS LET 51 4 MESSAGE 63 NEWLINE PROCESS 6 DONE
```

Now in process 1 we'll need to set both the players and his/her opponents strength levels to their initial values, using..

```
* * ZERO 31 LET 80 40 LET 81 40
```

Now, I've just set both strength levels to a limit of forty, which could always be added to with food, potions of strength, etc, or reduced by poisons, but they could be set to any value at all below 100, but not above 100, as you can only 'nest' a process upto a limit of ten, ie, you can only recall a process 10 times, before you get an error message. Now for process 6 in our second example, we could use..

```
* * 0 RANDOM 82

* * 1 CARRIED 10 PLUS 82 15
```

* * 2 CARRIED 11 PLUS 82 7

* * 3 ZERO 81 NEWLINE MESSAGE A SWAP 4 5 LET 80 40 ANYKEY DESC

Now you'll notice that the only real alteration I've made to this last routine, is to add a LET 80 40, to put the players strength back upto its initial value again, as the * * ZERO 31 action in process 1 won't be used again, and you've got to set the limits again yourself, or have the player have to find some way of renewing it him/herself, by food for example..

* * 4 ZERO 80 NEWLINE MESSAGE B NEWLINE TURNS NEWLINE SCORE NEWLINE END

You'll notice that most of the routines I've used so far (especially the last one above) are exactly the same as those in the first example..

* _ 0 GT 82 40 PROCESS 7 PROCESS 6

Now this time around, I've used process 7 as that which deals with any combat if the opponent is hurt..

* _ 1 LT 82 41 PROCESS 8 PROCESS 6

Just as the one above, but only if the player is hurt..

* _ 2 SAME 80 81 PROCESS 9 PROCESS 6

Same again, but only if neither the player or opponent is hurt, and now for these next three process tables, all three will need three routines within each, all exactly the same, but only the first two processes will use a minus action, and only the first two routines within each process will need a chance action, and thus for process 7 we'll need..

* _ 1 and 2 CHANCE 35 MINUS 81 A MESSAGE ?

* _ 3 won't need a chance action, with it being the last routine in the process, and all of these three are for the process when the attacker is hurt, so..

* _ 3 MINUS 81 A MESSAGE ?

Now for the routine above, and the following one, I've used flag values A and B to become a number from 1 to 20 (any higher and the odds are probably too much in the players favour, and he/she will probably win every time), the exact value I'll leave upto the writer, just like any names of characters and messages, and now onto process 8 (dealing with the resulting messages if the player is hurt)..

* _ 1 and 2 CHANCE 35 MINUS 80 B MESSAGE ?

* _ 3 MINUS 80 B MESSAGE ?

Now process 9 (our last fight process thankfully!) is the one dealing with any messages printed, if there's no harm caused to either the player or his/her opponent, needing no minus actions at all..

* _ 1 and 2 CHANCE 35 MESSAGE ?

* _ 3 MESSAGE ?

Well, that's it for another month!

The Ultimate Guide To GACing

By Matthew Conway

Part The First: Getting Started

Big deal, an article on how to get started with GAC. We've had enough of those as it is without some new berk telling us what we already know. Isn't that right?

Well, yes, it probably is, but then you'd be applying the wrong meaning to the word "started" if that's what you thought this article was all about. Don't worry, I'm not going to explain what a verb or noun is, nor am I going to describe how conditions work. Quite rightly, if that was my purpose then I'd deserve to be booed off the pages. However, it isn't, so shut up and listen!

"Getting Started" in this context refers to the beginning of the adventure as the player sees it, and the whole point of this article is to try and help you make it look as professional as possible given the tiny amount of memory the Spectrum has left after GAC has loaded. Inevitably, this is a lost battle if you try to do too much, but things can be kept short, sweet and, above all, simple.

The key to a good start is not to do the obvious and have the player immediately plunged into the first location. That's suicide because you will have wasted the perfect opportunity to get off on the right foot. After all, what could look better than an impressive title page proclaiming the adventure's name, the author, and anything else you wanted to be seen come to that? It certainly beats seeing "You are on a road" after waiting five minutes for the game to load.

The problem with GAC in this respect, however, is that this is exactly what the program wants to do. As the flowchart on the last page of the manual will show you, the very first thing that happens is that the player's location is described. This, you might think, immediately kills off any hopes you may have had of slipping something in before all this happens. Not so. In fact, this turn of events provides the perfect opportunity for a very easily-implemented title screen which takes up a miniscule amount of memory, allowing you to make it more impressive if you deem it worthy of the memory. Whatever type of introductory screen you plan to use, read on...

The key to this sneaky piece of programming is to make sure that the beginning location isn't a location at all! I know this will sound like rather an unusual idea to say the least, but it is this that lies at the crux of the whole routine.

I suppose a little elaboration is called for, so here it is. Quite simply, the location text of the room in which the player starts (best set to 9999 to avoid confusion when numbering the actual adventure locations) is the message which you want printed on the screen. Various other things can then be added onto this with the whole thing controlled through the use of a single High Priority Condition, before, finally, the press of a key takes the player to the first location of the adventure proper.

Good, isn't it?

The coding for this effect is equally simple, though, admittedly, it is difficult to conceive how it could be implemented before you've seen how it's done. I freely confess that I was stumped as to how the

message at the start of "Ransom", the free sample adventure with GAC, was put there until I bought a copy of The Essential Myth's "Gacpac" (highly recommended) and promptly proceeded to dismember the aforesaid program. It wasn't too hard to spot what was going on, and an amended version of it might go something like this:

```
Begin where? 9999

High Priority Conditions
Line 1:
IF ( RES? 4 ) SET 4 LP MESS 238 ROLD 65000 LP LP LP LP LP LP LP LP LP
     LP LP LP LP LP LP LP LP LP LP LP LP LP GOTO 1 END

Room description 9999:
Welcome to "Ransom", a small graphic adventure designed to show you
     some of the capabilities of GAC. In this adventure, you have to
     bring back some gold to the start to pay the ransom for your
     kidnapped king. Good luck! You'll need it...

Message 238:
Press any key...
```

Well, that's the code, but what does it all mean? Well, the Begin where? starts the game in 9999, a location which should never really be used and so will not affect the logical numbering of locations, ie: room 1 can still be the first location in the adventure proper. The description of room 9999 is the message you want printed on the screen, and message 238 is the one which tells the player to press a key to access the adventure. The real workhouse, however, is in High Priority Condition 1...

Things go like this. The current room description has already been printed - GAC does this for you - whether you are beginning in room 9999 or not. A check is then made on marker 4 (which should be left well alone during the rest of coding if you don't want silly things to keep happening): if it is set then you are not in room 9999 and the rest of the line is ignored, but if it is reset then you are in 9999 and the line will have its effect. Firstly, the marker is set to make sure that this line is ignored from here on - after all, you can't start twice in the same game, can you? Next, a blank line is printed, followed by "Press any key..." A pause is introduced to halt the program until a key is pressed (or until 21 minutes and 40 seconds have passed!) and then everything is scrolled off the screen to be replaced by the description of room 1 which is the first location proper. Easy, innit? (Note that the 23 LFs need only apply to introductory screens without an accompanying graphic - if there is one, allow two lines in the room description for the "Press any key..." message and then cut the number of LFs to 8.)

Of course, things don't stop there. This line is capable of handling any actions which need to take place before the player's first turn. Consequently, PICT or TEXT should be included depending on whether or not your adventure features graphics; SETs should be included for all markers which need to start off set and not reset (including marker 3 if you wish to disable GAC's built-in scoring mechanism); CSETs should be similarly included for counters beginning with values other than 0; and STRE should definitely be included to determine the maximum weight the player can carry at once.

Multiple messages can be placed on the title screen, though this is only really practicable if there is no accompanying graphic - after all, what can you really do in eight lines of text? Anyhow, if the screen is text-only then how about defining a few more messages and

placing them before the LF MESS 238 part of the Condition along with LFs to make them more readable? Or what about having more than one introductory screen, having several with each one being accessed in turn by the press of a key at which the current one is scrolled off the screen and the new one scrolled on? Careful use of LFs is needed to perfect this, but it is easily accomplished and still relatively easy-going on memory space. Finally, how about having a graphic introductory screen which scrolls away to reveal a screen full of text proclaiming the adventure title etc? This is very simple: merely insert the text for room 9999 into room 9998 instead, but still define the graphic as being for room 9999. Then change the High Priority Condition to:

```
IF ( RES? 4 ) SET 4 HOLD 65000 TEXT LF LF LF LF LF LF LF LF LF LF LF LF
     LF LF LF LF LF LF LF LF LF LF GOTO 9998 LF MESS 238 HOLD 65000
     LF LF LF LF LF LF LF LF LF LF LF LF LF LF LF LF LF LF LF LF LF
     LF GOTO 1 END
```

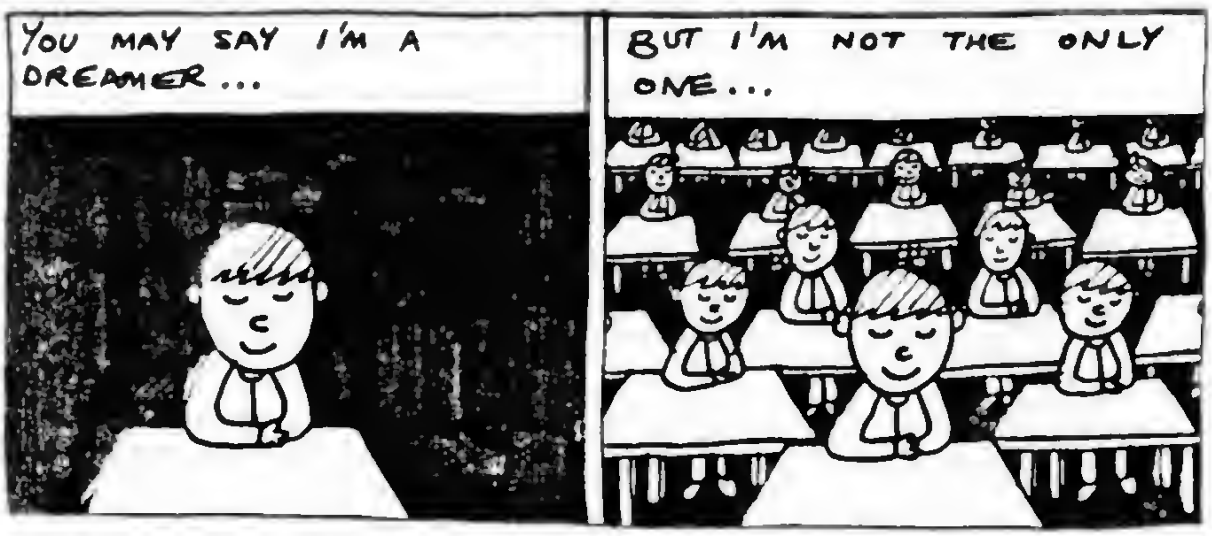
A lot of LFs I'm sure you'll agree and, in fact, you'll need a few more after the GOTO 9998 command to make sure that the text scrolls up to the top of the screen, but I'm sure that you'll agree that the result is certainly effective and extremely memory-effective to boot!

There you have it, then: a spectacular way to start off your GAC adventures. However, if you feel you have anything extra to add to this routine, or you have a GAC routine you feel could do with a public airing, or if you merely want to pass on a comment or ask a question about GAC, do feel free to write to me at the address below - but, please, if you want a personal reply outside of these pages then include an SSAE. It's so much cheaper for us poor writers!

Matthew Conway, 1 St George's Terrace, Station Road, Lambourn, Berks
RG16 7PW

The Dreamer

by Chris Hester



Machine Coding your adventures Part 2

by Paul Brunyee

Welcome to my second article on writing adventures using an assembly language. I hope to show in these articles how assembly languages can enhance the design of an adventure and give it an originality which a utility may not. However, bear in mind that the coding solutions I offer are not the be all and end all of coding. Simply put, there are an infinite number of ways to write a program to achieve a particular objective; my programs are just one of many possibilities.

I would urge people not to discount assembly languages for all their complexities. On the contrary, assembly languages are not at all difficult when the basic concepts are understood. It then becomes just another programming language.

In this article I will build on some of the ideas from Part 1 to create a command input routine, that routine which accepts a user's command (but doesn't act upon it). However, before writing any code, the routine must be clearly defined and the ground rules must be set out. The task is simplified a great deal by starting with a most basic version of the program in hand, rather than trying to cram every detail into the initial design. At a later date, when one version is fully working, and as your confidence and programming prowess improves, then the task of enhancing and improving the coding becomes easier. As any programmer can tell you, mistakes in an assembler program are often discovered in a rather 'final' way - the machine usually crashes or hangs, forcing a complete reload! With a high level language, such as BASIC, mistakes are dealt with interactively, e.g. the interpreter tells you at which line a 'division by zero' was attempted. This is not so with assembly languages, so be warned!

I'll set down the initial parameters as follows:

i) Commands will be restricted to one line.
ii) The only acceptable characters are A to Z and space.
iii) Commands will be entered on the bottom line.

This may, of course, conflict with other design criteria such as using separate windows, or having a multiple line input, but these enhancements may be added at a later time once the initial coding is satisfactory.

The approach to program design is an important consideration and has a direct consequence on the resulting code. Many people preach the need for 'pen and paper' work before touching a keyboard. This is often the case, but there are circumstances when this is not essential, mainly when using a high level language such as BASIC. Assembly language programming demands a high degree of planning because, as already mentioned, mistakes are costly. The approach I will take is to look at the objective as a whole, and gradually break it down into more manageable components, and then again break it down further into more definable areas until, as an end product of this refinement, assembler code exists.

Stage 1
Command input routine

Stage 2
Initialise.
Request input.

```
        Validate input.
        End routine.

Stage 2.1 - Initialise
        Set caps lock on.
        Set pointers (registers).

Stage 2.2 - Request input.
        Display prompt.
        Read keyboard.

Stage 2.3 - Validate input.
        Check for ENTER.
        Check for DELETE.
        Check for valid character (A to Z, and space).
        Store input.

Stage 2.4 - End routine.
        Set marker for end of input.
```

At this stage it should be easy enough to start coding. The coding occurs as the last logical step of the refinement process. The following code is a Z80 based code followed by routines specific for the Spectrum and Amstrad.

```
;Command input routine
;
        ORG 60000           assemble machine code at address 60000
        ENT 60000           entry point address
        CALL INIT           CALL 'initialise' routine
        CALL PRINTP         CALL 'print prompt' routine
        LD   E,1            horizontal cursor position for input
        LD   HL,INPUT       pointer to input store
LOOP1   CALL PRTSTP         CALL 'print full stop' routine
KEYLP   CALL GETKEY         CALL 'get keypress' routine
        CP   13             was it ENTER?
        JR   Z,ENDRTN       yes, jump to label ENDRTN
        CP   12             was it DELETE?
        JR   Z,DELETE       yes, jump to label DELETE
        CP   32             was it a space (CHR$ 32)?
        JR   Z,OK           yes, all is well
        CP   65             ASCII code for character 'A'
        JR   C,KEYLP        jump back if less than 65 (character 'A')
        CP   91             ASCII code for character 'Z' is 90
        JR   NC,KEYLP       jump back if greater than 90
OK      LD   B,A            save character code in reg. B
        LD   A,E            current cursor position (horizontal)
        CP   30             at end of line yet?
        JR   NC,KEYLP       yes, ignore input and jump back
        LD   A,B            put char. back in reg. A
        LD   (HL),A         store char. in input area
        INC  HL             increment pointer to input area
        CALL PRTCHR         CALL 'print character' routine
        INC  E              increment cursor position
        JR   LOOP1          loop back
DELETE  LD   A,E            current cursor position
        CP   1              trying to delete past prompt?
        JR   Z,KEYLP        yes, ignore input and loop back
        DEC  E              decrement cursor position (move left)
        DEC  HL             decrement pointer to input area
        JR   LOOP1          loop back
ENDRTN  LD   (HL),A         set CHR$ 13 as end of input marker
        RET                 return to system
```

```
INPUT   DEFS 31             define a space of 31 bytes

;
;Spectrum routines
;
INIT    LD    A,2
        CALL 5633           open channel 2 (to screen)
        LD    A,8
        LD    (23658),A     force caps lock on
        RET

;
;Print ">" at 21,0
;
PRINTP  LD    DE,PROMPT     point DE at characters to print
        LD    BC,4          length of string to print
        CALL 8252           CALL ROM print routine
        RET
PROMPT  DEFB 22,21,0        control bytes for PRINT AT 21,0;
        DEFM ">"            prompt character

;
;Print ". " at cursor position
;
PRTSTP  PUSH DE             temporarily save DE register pair
        LD    A,E           cursor position
        LD    (STOP1+2),A   put horizontal coordinate into control codes
        LD    DE,STOP1      point DE at string to print
        LD    BC,5          number of characters in string
        CALL 8252           CALL ROM print routine
        POP   DE            retrieve register pair from stack
        RET
STOP1   DEFB 22,21,0        PRINT control bytes
        DEFM ". "

;
;Read keyboard, put char. in Accumulator
;
GETKEY  RES   5,(IY+1)      reset bit 5 of FLAGS (system variable)
WAIT    HALT  ·             force KEYSCAN routine to read keyboard
        BIT   5,(IY+1)      test bit 5 of FLAGS
        JR    Z,WAIT        loop back if no keypress
        LD    A,(23560)     load keypress code into Accumulator
        RET

;
;Print character in reg. A at cursor position
;
PRTCHR  PUSH AF             save character for the time being
        LD    A,22          control code for PRINT AT
        RST   16            print character in Accumulator
        LD    A,21          control code for row 21
        RST   16
        LD    A,E           horizontal cursor position
        RST   16            PRINT AT 21,E;
        POP   AF            retrieve AF register pair
        RST   16            print character in Accumulator
        RET

;
;Amstrad routines:
;
INIT    CALL VDUENA         allow chars to be placed on screen
```

```
        CALL CURDIS       prevent cursor blob display
        RET

;
;Print ">" at 25,1
;
PRINTP  LD   H,1          column 1
        LD   L,25         row 25
        CALL SETCUR       set cursor position to 25,1
        LD   A,">"        load A with ASCII value for ">"
        CALL OUTPUT       print char. in A at current position
        RET

;
;Print ". " at current position
;
PRTSTP  PUSH HL           save HL for time being
        LD   A,E          horizontal cursor position
        INC  A            increment A for logical screen position
        CALL SETCOL       set horizontal cursor position
        LD   A,"."        load A with ASCII value for a full-stop
        CALL OUTPUT       print char. at current position
        LD   A,32         load A with ASCII value for a space
        CALL OUTPUT       print space following the full-stop
        POP  HL           retrieve HL register pair
        RET

;
;Read keyboard, put char. in Accumulator
;
GETKEY  CALL WAITCH       read char. from keyboard into A
        CP   91           check for upper/lower case
        RET  C            RETurn if upper case
        CP   123          is it a lower case character?
        RET  NC           no, RETurn if code >= 123
        SUB  32           turn char. into upper case
        RET

;
;Print char. in A at cursor position
;
PRTCHR  PUSH HL           save HL for time being
        PUSH AF           save AF for time being
        LD   A,E          horizontal cursor position
        INC  A            increment for logical screen
        CALL SETCOL       set horizontal cursor position
        POP  AF           A contains char. to print
        CALL OUTPUT       send char. to output routine
        POP  HL           restore HL register pair
        RET

;Amstrad EQUates
;Equate label with a hardware address (in hex) to branch to:
;
VDUENA  EQU  &BB54        TXT VDU ENABLE
                          allow characters to be placed on screen
CURDIS  EQU  &BB7E        TXT CUR DISABLE
                          disallow cursor display
SETCUR  EQU  &BB75        TXT SET CURSOR
                          set cursor position
OUTPUT  EQU  &BB5A        TXT OUTPUT
                          output character to Text VDU
```

```
SETCOL  EQU  #BB6F      TXT SET COLUMN
                       set cursor horizontal position
WAITCH  EQU  #BB06      KM WAIT CHAR
                       wait for next character from keyboard
```

I wish to point out at this stage that the preceding Amstrad routines were gleaned from a colleague's Firmware manual and I have not been able to test them, therefore I cannot state that they will do exactly as required. If anyone can shed any light on the accuracy of these routines then I would be pleased to hear from you.

I have also taken into account the fact that the Amstrad logical screen difers from the Spectrum's as the leftmost column is column 1, as opposed to the Spectrum's column 0. Furthermore, the DELETE code for the Amstrad may need to be changed to #7F. Can anyone advise? The Spectrum routines, however, have been tested and function correctly.

Next moon I will move onto the topic of command parsing and present designs and coding for this. Please forward any querres or comments you have and I will help as best I can.

Whatever happened to..?

This column is about unreleased programs - what happened to them?

Hear this!

"Click, clack, click, clack, click, thud. Knock, knock, silence. Bash, creak... bash, creaaaaakkk, slam. Phew! Click, clack, click, ching! Ugh..., ching, fup. Click, clack, roar. Click, roar!, clack, ROAR, aaaarrrgghhh!!!

End of game. Any key to try again."

The above is a sample from an imaginary playing of the world's first sound-only game. There's no text whatsoever at all. You don't read "click, clack" on the screen, you *hear* it, as the sound of your footprints moving in the direction you choose from the keyboard. It's assumed you're wearing hard shoes exploring an underground labyrinth, hence the sound of your heels on the stone you walk along. But there's no graphic to illustrate the scene - the only picture you get is in your mind, entirely generated from the sounds you're hearing from the computer!

In the above sample game, you walked several steps forward until you banged into a door - "thud". From there, you tried knocking, but to no avail, so you tried bashing the door down with your fist. The door responded by creaking open, bit by bit! Gradually it swung right open and slammed against the wall ahead. You continued walking until you hit a key with your shoes on the ground - "ching". You bent down and picked it up, putting it in your backpack. All seemed well as you carried on walking until you could hear the sound of a roaring animal. You foolishly kept on as the roar increased in volume. Too late! The roar deafened you as you were attacked by a hideous monster from behind the next corner. You couldn't see it hiding, but you heard it roar.

An adventure game? Not as we know it! All adventures rely on using either text and/or graphics to depict their actions. The screen is therefore crammed with something, but imagine if it were completely blank - jet black. This is exactly what you'd find in the sound-only game (SOG) I'm describing here. All you have to play it is the keyboard and the sound-effects. You don't type in words, the keys are defined instead to represent major actions used within an adventure game, such as GET AN OBJECT or MOVE NORTH. It's a bit like "Lords Of Midnight" except there's no view! With a blank screen, there's only sound to describe your situation. You know you're in a corridor if your feet echo as they click along what has to be a stone floor. You know you're at a waterfall if you can hear the sound of rushing water. You know you're being attacked if you can hear a screaming monster. There's all you need in the sound-effects to play a SOG, but would it work, and why?

Well, for a start, why not? If you kept the puzzles simple, such as finding keys ("ching") to open doors ("thud") and swords ("clunk") to attack monsters ("roar") and so on, then there'd be no problem with representing each aspect of your sound-only game with a sound-effect. For realism, these could easily be sampled, and the samples altered in pitch to create different objects - a sword and a dagger might be represented by the same sample of a metal object dropping, but played back accordingly. At a high pitch the sample might represent the dagger - "clink". Played back at a lower pitch, it might suggest a sword - "clunk", and lower still, a great heavy axe or something similar - "clonnkk!". All from the same sample. With a basic set of samples, you'd be able to rapidly construct your entire objectry -

does such e word exist? - plus pleyer's ections. This would teke up s lot of memory, but there'd be e lot of memory free enywey, since you'd be using no text! Sound-effects would eliminete the need for object desciptions, and iocetions would be depicted es you imegine them in your mind. Great! This would setisty people who complain that graphics in adventures take ewey the individuel's unique visuel interpretetion of a geme. Even text limits e scene to the wey the text is written - "e corridor" sounds very dull, and "e lengthy corridor of encient stone" conveys more atmosphere, yet it still wouldn't be es atmospheric as *hearing* the corridor from your footsteps es they echo down it! And you'd know it wes a corridor, not e room, from the simple wey it progressed lineerly on your mep. Meking e mep would be essentiel of course in e SOG es you wouldn't be eble to see where you were. Thet wouldn't meke it iapossible though, ell it would meen is thet you'd heve to try eech direction to heer if you could move thet way or not. In our exemple corridor, let's essume it runs north for ebout ten steps. with e door helfwey. Depending on the computer you'd be using, different keys would represent your directions, probably indiceted by e keyboard overley to help you, es found in the peckeging of "Lords Of Midnight". On e typicel keyboerd, you could use the cursor keys to represent directions, so the key to move the cursor left would give you the equivalent of WEST, end the cursor-up key NORTH. You might even be eble to use the joystick. In our corridor, pressing oursor-left tries to teke us WEST, but the corridor only runs northwards, so you'd heer "bump" - you've hit e well! You might heve to work this out yourself, or the instructions for the geme might indicete this es I heve. Try cursor-right then to move EAST - "bump". If you got "thud" instesd, then you'd heve hit a door. Cursor-up then? "Click" - greet, you've moved north one step. Agsin - "clsck", snother step, end so on!

At the end of the geme, the computer responds with some sempled speech - "End of gsme". Besides thet, the game would be entirely represented with sound-effects, end meybe e bit of music in the bsckground too. Think I'm crazy? Well think egain, for this is exectly the type of gsme once proposed for relesse!! I em pretty sure that it wes CRL's Nu Wave lebel thst ennounced e few years back to produce the first SOG. I remember feeling excited et the prospect - whst would it be like? Would the screen be blenk? Could you actually pley it? Ales, since this is e column devoted to unreleesed programs you can guess the fete of Nu Weve's SOG. Not e thing was heerd ebout it ever since. One minute they were going to releese it (imegine e review!), the next nothing.

Besides the novel etmospheric velue of e sound-only geme, it might heve actuelly been immensely useful to blind people. They cennot reed eny text, so edventures are completely out for them, es ere ell types of computer gsmes... except a SOG! Just think, blind people would heve no problem with understanding one - they wouldn't need to resd eny text as all ection is described in sound form. Wonderful! The only problem would occur in using e keyboard, femiliarising themselves with the leyout of the keys - they'd have to leern which keys did what end remember it all. However, e joystick would be useful for directions. end maybe even other ections if the fire button is used - for exsmple, UP and FIRE et the same time on the joystick might mesn GET AN OBJECT. whstever. I hsven't tried it, but you might be able to get ALL the ections vie the joystick snd not heve to use the keyboard et sll - idesl for blind people again! Just think - a SOG would be the first computer geme they would be eble to pley, end think of all the ores a sighted person can choose to pley. Just one would be of immense satisfaction to the blind. For that reason elone I feel disappointed that Nu Wave, if it were them, never got round to finishing their unique SOG idee. Anyone out there willing to tske up the chellenge!

A Whole STAC Of Problems

By Matthew Conway

Part The First: Special Condition 17 And All That Jazz

This is the start of my writing career for "Adventure Coder" and so, before sitting down and typing this article, I thought to myself, "What shall I do to start with?" Of course, it didn't take me long to work out the answer - it's quite obvious really: I shall start at the start!

The start in any STAC adventure is determined by what code is put into Special Condition 17. However, this can't just be any old code because S17 (as I shall call it to save valuable ink and sanity) determines the very first things that the player will see of the actual adventure. A loading screen is all well and good, but the most it can do is to leave the player with a sense of awe at the artist's graphical prowess. Such a feeling, however, is grossly detrimental if the next thing to come up before his/her eyes is an awefully-designed introductory screen which appears to have been given all of ten seconds' thought and a similar amount of time in actual programming. As anyone who has ever reviewed an adventure will know, this is as bad a start as can be imagined and ratings are invariably affected, irrespective of the adventure's true virtues. First impressions are vital, and this article is designed to try and help you get the most out of S17.

Before getting down to the code itself, it is vital that you make a list of everything that you are going to put into S17. This may seem quite unnecessary because, after all, if you're clever enough to write an adventure then surely you must be able to remember what you are going to put into that adventure, right? Wrong! Try to do this and you will invariably find that the little ideas which make a good game a great game begin to slip your mind, and there's nothing worse than knowing that you had a great routine just begging to be used and then forgetting what it was. Whenever you get a good idea, scribble it down on a scrap of paper so that you won't forget it. Believe me, having a permanent record of that routine will soon prove to be a blessing and not the waste of time which it initially seems to be.

Anyway, back to S17. This condition must contain a) anything which is set up at the start of the adventure and b) anything to be printed on the screen before the adventure proper begins.

To elaborate, a) includes the initial colour scheme, the screen mode depending on whether the game is text-only or incorporates graphics, the cursor size, counters initially set to values other than 0, markers initially set and not reset, the player's strength value etc, while b) includes the title screen, request for instructions and background etc.

Okay, so you've made a list of what you want to incorporate into S17. What next? Well, now comes the time for turning all that English into something STAC can understand. Don't worry, though, because I'm not going to advise that you do this on paper first and then transfer it all to the computer - this is one time when putting everything straight into STAC works. However, do take care that you do things one at a time. Never move onto the next item on your list until you have fiddled and tweaked with the previous one to your total satisfaction. Another thing to do is to always place separate ideas on different lines. There is no advantage in lumping the entire condition onto one line and then realising that you can't work out where one item ends and the next one starts - lay things out logically and clearly and you will never have any problems. If need be, take advantage of the fact that you can place comments on these lines to label them - there is absolutely no excuse

for forgetting the purpose of a line of code if you do this!

As an example of what I think a well laid-out and effective S17 looks like, here is one of the sort that usually adorns the start of any adventure I might end up writing, along with comments of what each part of the code does:

```
setstr 100
0 colour 777 3 colour 0 0 topcol 777 3 topcol 0
text
split
message 1 lf lf message 2 lf lf message 3 lf lf message 4 lf lf lf lf
     message 5
if yesno then special 19
```

What this all does is this:

- setstr 100 - set the player's strength value to 100 units, a convenient number for expressing objects' weights as a percentage of what the player can carry.
- 0 colour 777 etc - set the initial colour scheme to an all-white background with black text, the easiest to read even if it is somewhat uninspiring.
- text - I rarely work with graphics and so consequently the screen mode is set to text-only.
- split - switch to 80-column mode, perfectly usable if a legible font is created.
- message 1 etc - print the name of the adventure, where the idea came from, who wrote it, the copyright notice, and ask the player if he/she would like to read the background to the adventure.
- if yesno then special 19 - if the player wants to read the background, jump to Special Condition 19 which facilitates just that.

Of course, there are very many other things which could be done in S17. As you begin to implement more and more complex routines, new initial values have to be set. For example, if you wish to implement a routine to force STAC to print "and" before the last object in a room location (a routine I shall be writing about in the near future!), a series of mess@ commands have to be placed in S17. Other things crop up all the time - you notice that you need to preset the value of a counter, so that goes in S17. A marker needs to be set before the game starts - S17 again. The range of usages for S17 is enormous, so make sure you make the best of it.

Before this article comes to an end, a few words about what makes a good adventure a great adventure are probably in order, connected as they are with the whole idea of starting things off.

Most importantly, though I hate saying it and purists will probably die of sudden seizures, is the inclusion of graphics. If you have any hopes of getting an adventure picked up by a top software house, graphics are 99% essential. Unfortunately, only a select group of aged adventurers who have been around since the days of "Colossal Adventure" find text adventures to be the best sort there is - the modern adventurer, demanding his machine to be stretched to the limits and his eyes to be assailed by all sorts of graphical delights, believes quite the opposite. Which all goes to indicate that I'll never get rich through this programming lark.

Anyway, personal problems aside, a decent plot comes next. Originality is not as important as some would have you believe, but a credible background which excites the player's imagination is. If you can carry

thie on into the adventure, as much fun can be gained by exploring the game world as from trying to solve the probleme which abound in it.

Thirdly, do not, under any circumstancee, inelude sudden death routinss. Never, never, never. No way, no how. Sudden death in an adventure means sudden death for an adventure. There is nothing more infuriating than eolving a really vicious problem only to be killed becsuse you forgot about the large monster to the north which you weren't told about about and which gives you no chance of eecape. Avoid.

Finally, spare no expense at making the parser as responsive as poesible. "You can't" helps no-one. If there is some reason given as to why ths player can't do what he/she just typed in then he/ehe is more likely to plsy on than just eit back and give up. Nobody expects a STACed game to respond like an Infocom adventure, but the nearer you get to that level then the nearer you are to producing a very good game indeed.

Well, that just about wrape up thie little discourse on how to get the most from the start of an adventure. If anyone hae any comments they would like to mske about what I have said here, or if anyone would like to send me any routinee which they think would benefit the rest of the readerehip, or if anyone hae a query about an aspect of STAC with which they are not totally at home, then just drop me a line. Howsver, if you would like a pereonal reply outside of these pagee, pleaee eend an SSAE ee my postal eosts are high enough ae it is!

Matthew Conway, 1 St George's Terrace, Station Road, Lambourn, Berks RG16 7PW

Back Issues

These are avsilable at the ssme price as s normsl issue (see bsck page for full detsils sbout prices).

ISSUE 1 July 1989

GAC+ review! PAW Prints! Machine Coding your sdventures Part 1! Whatever happened to... "Vslley Of The Source"! GA. graphics article on Colour, Perspective, Ellipses snd Rectsngles etc! Fiction - "The Burning Man"!

The Adventure: how to write one

by Patrick Walsh of Mental Image

Preliminary

Make sure you have read the manual of your adventure system and understood it as fully as possible. Obviously if you don't understand the system it will cause you a lot of frustration later on, when you should be concentrating on writing your adventure.

The storyline

Hopefully you will already have an idea; maybe from a book you have read, a newspaper story, a film or many other sources. If you haven't already got an idea then these are quite difficult to come by, especially a good one. You will be more likely to suddenly hit upon an idea, rather than sitting down trying to get an idea.

Develop your idea into a proper stoyline, with the main things that will happen in your adventure. Try to establish some main objectives, with smaller objectives leading to and from these. Make sure you have an overall objective for the player to aim for, and make sure it is interesting enough for the player to continue to play.

Make a plan of the locations you will have in your adventure, along with a map of all these locations. Once you have decided the tasks necessary to complete the game you should have a number of objects, which can be added to as you wish. I like to make a list of all the objects I am using in my game, along with their numbers, all the nouns used and their noun number. I find it best to write these lists on a piece of A4, to keep them all together and to allow for additions. I also write the message number of each examinable object next to its decription, this helps tracing messages to their source. On the same sheet of paper I also write each flag, it's number and what it does. This is vital, as without it you would be lost when debugging.

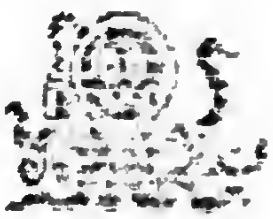
The writing

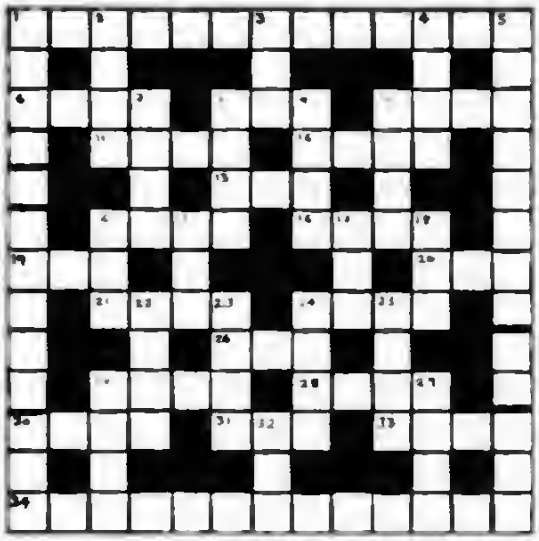
With your A4 sheet, map and storyline at hand, plus a large writing pad you can start writing your adventure. I would suggest that you go through your adventure writing and completing one location at a time, before going on to the next one. I find this helpful, as if you keep jumping from place to place you will soon get confused.

If you plan to use PSIs (characters) I would suggest that you write down what each one will do; where it will go, if you can talk to it; does it talk to you, does it grab objects etc etc.

Once you have finished your writing you will need to rigorously playtest and spell check your adventure. It does help if you playtest each location as you complete it and this will leave much less work at the end. It would be helpful if you could send finished copies to friends who can playtest it for you, and suggest new ideas.

Then comes the really hard bit, trying to sell your game. I will leave that till another time.





Crossword #2

ACROSS

1) Metamorphosis (13)
6) Falling water (4)
8) Plus (3)
10) Long for (4)
11) Close to the heart (4)
12) Great musical work, often numbered (4)
13) Charged particle (3)
14) Reared (4)
16) Always (4)
19) Throw hard (3)
20) Donkey (3)
21) Duplicate (4)
24) Religious group (4)
26) Long time period (3)
27) Place (down) to rest (4)
28) Join (4)
30) Not odd (4)
31) Solid water (3)
33) Animal skins (4)
34) Remarkable (13)

DOWN

1) Ignore (4,1,5,3)
2) Keen (4)
3) Vase for ashes (3)
4) Circular part of eye (4)
5) Obviously (8,2,3)
7) Close (4)
8) Dry (4)
9) Finished (4)
10) Unspoilt (4)
14) Major television and radio company (1,1,1)
15) Sixth sense? (1,1,1)
17) Strive for superiority (3)
18) Roland --- (3)
22) Cooker (4)
23) Abominable snowman (4)
24) Identical (4)
25) Cook (4)
27) Posted (4)
29) Fish used for paste (4)
32) Motor vehicle (3)

Devised by C Hester

The solutions to both crosswords this month will appear in the next issue No.3. September 1989.

Have you devised a crossword yourself? Or perhaps a puzzle of some kind - if so, why not send it in?

Useful addresses

This list is intended to help you in selling a game and getting it revieved. If you have any other addresses you've found useful in the past, let me know and I'll include them in future issues.

AMI = Amiga
ARC = Archimedes
BBC = Acorn BBC Micro
C64 = Commodore 64/128
CPC = Amstrad CPC range
ELE = Electron
S48 = Spectrum range
ST = Atari ST range
VAR = various computers

ADVENTURE MAGAZINES

VAR: Adventure Coder:- Christopher Hester, 3 West Lane, Baildon, Nr. Shipley, West Yorks, BD17 5HD.

VAR: Claus Nygaard, Adventure Posten, Adventure Klubben, Vestergade 25A, 4930 Maribo, Denmark.

VAR: Mandy Rodrigues, Adventure Probe, 24 Maes Y Cwm, Llandudno, Gwynedd, LL30 1JE.

S48 tape: Magic Missile, Futuresoft, 75 Ben Rhydding Road, Ilkley, West Yorkshire, LS29 8RN.

VAR: Mike Brailsford, Spellbreaker, 19 Napier Place, South Parks, Glenrothes, Fife, KY6 1DX.

ST disk: Syntax, 9 Warwick Road, Sidcup, Kent, DA14 6LJ.

ADVENTURE COLUMNISTS

VAR: Steve Cooke, Ace, Priory Court, 30-32 Farringdon Lane, London, EC1 3AU.

AMI: Dave Eriksson, Amiga Computing, Database Publications Ltd, Europa House, Adlington Park, Adlington, Macclesfield, SK10 4NP.

ST: Brillig, Atari ST User, Database Publications Ltd, Europa House, Adlington Park, Adlington, Macclesfield, SK10 4NP.

C64: Andy Moss, Commodore Computing International, Croftward Ltd, Finsbury Business Centre, 40 Bowling Green Lane, London, EC1R 0NE.

C64: Gordon Hamlett, Commodore Disk User, Argus Specialist Publications Ltd, Argus House, Boundary Way, Hemel Hempstead, HP2 7ST.

C64/AMI: keith Campbell, Commodore User, Priory Court, 30-32 Farringdon Lane, London, EC1 3AU.

VAR: keith Campbell, Computer + Video Games, Priory Court, 30-32 Farringdon Lane, London, EC1 3AU.

ELE: Pendragon, Electron User, Database Publications Ltd, Europa House, Adrington Park, Adlington, Macclesfield, SK10 4NP.

VAR: Paul Rigby, The Games Machine, PO Box 10, Ludlow, Shropshire, SY8 1DB.

BBC: The Mad Hatter, The Micro User, Database Publications Ltd, Europa House, Adlington Park, Adlington, Macclesfield, SK10 4NP.

VAR: Tony Bridge, Popular Computing Weekly, Greencoat House, Francis Street, London, SW1P 1DG.

S48: The Sorceress, Sinclair User, Priory Court. 30-32 Farringdon Lane, London, EC1 3AU.

S48: Mike Gerrard, Your Sinclair, 14 Rathbone Place, London. W1P 1DE.

C64/AMI: Prof Norman Nutz, ZZAP!, PO Box 10, Ludlow, Shropshire, SY8 1DB.

ADVENTURE COMPANIES

VAR: Alternative Software Ltd, Units 3-6, Baileygate Industrial Estate. Pontefract, West Yorkshire, WF8 2LN. Telex: 557994 RR DIST G Fax: (0977) 790243 Tel: (0977) 797777

VAR: Digital Dynamite, 54 Watermill Road, Fraserburgh, Grampian, Scotland, AB4 5RJ.

VAR: Mitch Pomfret, M.S.B Games. 2 Bude Close, Bramhall, Stockport, Cheshire, SK7 2QP. (GAC)

VAR: Mastertronic, 2-4 Vernon Yard, Portobello Road. London, W11 2DX.

VAR: Rack-It, Hewson Consultants Ltd, 56B Milton Park, Abingdon, Oxon, OX14 4RX. Tel: (0235) 832939

S48: John Wilson, Zenobi Software, 26 Spotland Tops, Cutgate, Rochdale, Lancashire, OL12 7NX.

ADVENTURE UTILITIES AND/OR ADD-ONS

S48: Camel Micros, Wellpark, Willeys Avenue, Exeter, Devon, EX2 8BE.

CPC: Roger Bankin, Graduate Software, 14 Forrester Avenue, Weston on Trent, Derbyshire, DE7 2HX.

VAR: Incentive Software Ltd, Zephyr One, Calleva Park, Aldermaston, Berkshire. RG7 4QW. Tel: (07356) 77288 Fax: (07356) 6940

VAR: Gilsoft International Ltd, 2 Park Crescent, Barry, South Glamorgan, CF6 8HD. Tel: (0446) 732765

S48: Gerald Kellett, kelsoft, 28 Queen Street, Stamford. Lincolnshire, PE9 1QS.

CASSETTE DUPLICATORS

JBS Records, Freepost, 19 Sadlers Way, Hertford, SG14 2BR.

McGregor Tape Services, 42 Anchor Avenue. Paisley, PA1 1LD.

Simon Stable Productions. 20 West End. Launton, Oxon, OX6 0DF.

STATIONERY, PACKAGING AND PRINTING

Launton Press Ltd, Wedgewood Road. Bicester. Oxon.

Millway, Chapel Hill, Stanstead, Essex.

S&M (Processing) Ltd, Gotts Road, Wellington Bridge, Leeds, LE12 1BS.

Utilities and add-ons

If you know of any other utilities or add-ons, especially for computers such as the MSX and Atari 8-bits, whatever, please write in and help make this list a definitive guide.

AMI = Amiga
CPC = Amstrad CPC range
ARC = Archimedes
BBC = Acorn BBC Micro
C64 = Commodore 64
DRG = Dragon 32
ELE = Electron
MTR = Master
S48 = Spectrum 48K
ST = Atari ST range

PROGRAM NAME	COMPANY (COMPUTERS) COMMENT
A-CODE	Level 9 (many) in-house utility only
ADLAN	Graduate (CPC)
ADVENTURE BUILDER SYSTEM	M A Richards (S48)
ADVENTURE CONSTRUCTION SET	Electronic Arts (C64)
ADVENTURE KERNEL SYSTEM	Melbourne House (AMS) book listing/tape
ADVENTURE WRITER	Codewriter (C64) USA Quill
ADVENTURESCAPE	A&B (BBC)
ADL	Public Domain (AMI)
ADVSYS	Public Domain (ST)
ALPS	Alpine Software (BBC MTR ARC)
AMIGAC?	Incentive (AMI) Coming soon?
AMIGAVENTURE	Public Domain (AMI)
THE BIRO	Ramjam Corporation (many) in-house/to loan
CHARACTER SETS	Simicro (S48) GAC
CHARACTERS	Gilsoft (S48) Quill
DRAGON WRITER	Cowen (DRG)
DUNGEON BUILDER	Dream (C64)
THE EXPANDER	Gilsoft (S48) with PRESS
FONT CREATOR	Simicro (S48) GAC
THE FIX	Kelsoft (S48) Quill
THE FIX+	Kelsoft (S48) Quill - unreleased
GAC	Incentive (S48 AMS C64)
GAC+	Incentive (C64) disk-only
GAC DATABASE PRINTER	Big Sky (C64)
THE GACPAC	Essential Myth (S48) GAC
GENESIS	CRL/Camel Micros (S48) good band!
THE ILLUSTRATOR	Gilsoft (S48 AMS C64) Quill
MEGA	Gilsoft/Kelsoft (S48) PAW, part of PTM
MINIFIX	Kelsoft (S48) Quill
PATCH	Gilsoft (S48) Quill
PAW	Gilsoft (S48 AMS PC) no C64/ST!
PAW-PHOSIS	Gilsoft/Kelsoft (S48) PAW, part of PTM)
PAW-TEL	Gilsoft/Kelsoft (S48) PAW, part of PTM)
PRESS	Gilsoft (S48) Quill
PTM	Gilsoft/Kelsoft (S48) 3 PAW overlays
QUAID	Kelsoft (S48) Quill
THE QUILL	Gilsoft (S48 AMS C64)
RECLAIMER	Kelsoft (S48) GAC
SAGA	Scott Adams (C64) not for sale!
THE SCRIBE	Your Spectrum (S48) listing
STAC	Incentive (ST)
TAC	Incentive (BBC ELE) GAC without graphics
TAILSPIN	Microdeal (ST AMI)

SUBSCRIPTIONS

Price per single issue:

£1.00 United Kingdom
£1.75 Europe (+ S.Ireland)
£1.75 Rest of the world (surface)
£2.25 Rest of the world (air mail)

Price per 12 issues (1 year):

£12.00 United Kingdom
£21.00 Europe (+ S.Ireland)
£21.00 Rest of the world (surface)
£27.00 Rest of the world (air mail)

All payments must be made in British sterling money. Cheques or postal orders should be crossed and made payable to "C HESTER" please. (This also applies to the costs below.)

ADVERTISING RATES

£5.00 Inside full page (black-and-white A5)
£9.00 Inside double-page spread (middle pages)
£6.00 Outside back cover (on coloured paper)
£2.50 Inside half-page
£1.25 Inside quarter-page

Line ads cost only 3p per word.

LETTERS

Letters sent to the magazine may be chosen for printing unless marked "Not for Publication". If you require a reply other than printed in the magazine, please enclose a stamped self-addressed envelope.

COPYRIGHT

ABOUT ADVENTURE CODER

The opinions expressed within this magazine by individual writers are not necessarily shared by the Editor. Adventure Coder does not recommend or condone any particular firm or product.

Issues of Adventure Coder come out during the middle of each month. The deadline for contributions is the 25th of the previous month.

Software and hardware is greatly appreciated for review purposes no matter what computer it is for. Adventure writing utilities are especially welcome.

Adventure Coder is produced on a Commodore 64 with a Citizen 120D printer using a variety of programs.

PUBLISHER: Mandy Rodrigues

EDITOR: Christopher Hester, 3 West Lane, Baildon, Near Shipley, West Yorkshire, BD17 5HD, England.